REMARKS

In the Office Action, the Examiner indicated that Claims 1 through 20 are pending in the application and the Examiner rejected all claims.

Claim Rejections, 35 U.S.C. ' 102 and ' 103

On page 2 of the Office Action, the Examiner rejected Claims 1-2, 4-9, 11-16, and 18-20 under 35 U.S.C. §102(e) as being anticipated by "Secure Java Class Loading", IEEE Internet Computing, November/December 1998, Pages 56-61 by Gong ("Gong"). On page 8 of the Office Action, the Examiner rejected Claims 3, 10 and 17 under 35 U.S.C. §103(a) as being unpatentable over Gong in view of U.S. Application Publication No. 2003/0115218 to Bobbitt et al. ("Bobbitt").

The Present Invention

The present invention is a method, system, and computer program product for compiling Java code. In accordance with the present invention, Java code that references classes residing in a workspace can be compiled. In accordance with the present invention, a workspace identifier is placed within the classpath to indicate the location of the referenced classes that may reside within a workspace. Specifically, Claim 1 recites: "1) determining if a referenced class file is located in a workspace; 2) locating the class file; 3) accessing the class file; and 4) returning the class file data to the compiler wherein said compiler executes said class data file to produce machine executable code without removing any class data files from said workspace."(lines 3-8) In accordance with a preferred embodiment, the files on a web site are serviced using a file database, and a class file is allocated to a workspace by creating an additional entry in the file database. The class file is invoked by the compiler through the database and processed without moving the location of the class file in the workspace.

<u>"Secure Java Class Loading", IEEE Internet Computing, Nov/Dec 1998, Pages 56-61 by Gong</u>

"Secure Java Class Loading", IEEE Internet Computing, November/December 1998, Pages 56-61 by Gong ("Gong") teaches dynamic class loading as a feature of the Java virtual machine. Gong defines several unique characteristics of class loading, including loading classes on demand, on a just-in-time basis. Second, dynamic class loading maintains type safety of the Java virtual machine by adding link-time checks, which replace other runtime checks. Additionally, dynamic class loading provides programmers with additional functionality, including defining their own classes, and the ability to utilize class loaders to provide separate name spaces for various software components. The Examiner acknowledges that Gong fails to disclose an indicator comprising a signature string, a user ID, a project ID, and a workspace name.

<u>The Cited Prior Art Does Not Anticipate the Claimed Invention</u>

The MPEP and case law provide the following definition of anticipation for the purposes of 35 U.S.C. §102:

> AA claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.@ MPEP ' 2131 citing *Verdegaal Bros. v. Union Oil Company of California*, 814 F.2d 628, 631, 2 U.S.P.Q. 2d 1051, 1053 (Fed. Cir. 1987)

<u>The Examiner Has Not Established a *Prima Facie* Case of Anticipation</u>

By this amendment, Applicant has modified independent claims 1, 8 and 15 to further define the present invention as novel over the prior art of record. Specifically, the claims were amended to include accessing and compiling source files associated with class files stored in a workspace *without moving the class files from the workspace*. This reduces the computational overhead of the system as duplication of files is eliminated. Also, the potential for a user to accidentally move a file another user needs is eliminated. This step is not taught or suggested by the prior art.

Gong teaches a Java virtual machine (JVM) using a classpath and ClassLoaders to execute a bytecode in the JVM, or essentially, running Java applications, by copying and loading any required class files into a local memory running the JVM for execution. This differs greatly from the presently claimed invention. Specifically, the present claimed invention locates a class file, accesses a class file, and virtually returns the class file to a compiler for further processing without moving the location of the class file or copying the class file. Gong specifically functions by loading classes into local memory, that is, it moves them from the workspace. In the Office Action, page 6, the Examiner clearly defines class loading and its use in Gong, specifically, that they "specify the remote location from where certain classes are loaded" and are copied for processing by the JVM.

Accordingly, each of the independent claims, and all claims depending therefrom, patentably define over Gong and are in condition for allowance. The Examiner is respectfully requested to reconsider and withdraw the rejections of Claims 1-2, 4-9, 11-16, and 18-20 under 35 U.S.C. §102(e) as being anticipated by Gong.

Conclusion

The present invention is not taught or suggested by the prior art. Accordingly, the Examiner is respectfully requested to reconsider and withdraw the rejection of the claims. An early Notice of Allowance is earnestly solicited.

The Commissioner is hereby authorized to charge any fees associated with this communication to Deposit Account No. 09-0461.

Respectfully submitted,

March 31, 2008
Date

/Mark D. Simpson/
Mark D. Simpson, Esq.
Registration No. 32,942

SYNNESTVEDT & LECHNER LLP
1101 Market Street
Suite 2600
Philadelphia, PA 19107
Telephone: (215) 923-4466
Facsimile: (215) 923-2189
S:\I\IBM\IBM RALEIGH RSW\PATENTS\P27088 USA\PTO\REPLYTOOA12312007.DOC